НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

ЗВІТ

про виконання лабораторної роботи №2

з дисципліни «Чисельні методи математичної фізики»

на тему

«Задачі на деформацію пластини»

Варіант 1

|  |  |
| --- | --- |
| Виконав: | Перевірив: |
| студент групи КМ-11мн | Проф. |
| Агафонов Д. С. | Ориняк І. В. |

Київ — 2021

ЗМІСТ

[1 ПОСТАНОВКА ЗАДАЧІ 2](#_Toc92160036)

[2 АНАЛІТИЧНІ РІШЕННЯ 3](#_Toc92160037)

[2.3 Метод Нав’є 6](#_Toc92160038)

[3 МЕТОД КРАНКА-НІКОЛСОНА 9](#_Toc92160039)

[3.1 Різницеві схеми 9](#_Toc92160040)

[3.2 Алгоритм реалізації МКР 11](#_Toc92160041)

[Група 1. 12](#_Toc92160042)

[Група 2. 12](#_Toc92160043)

[Група 3. 12](#_Toc92160044)

[Група 4. 12](#_Toc92160045)

[3.3 Візуалізація результату 12](#_Toc92160046)

[Додаток А. Лістинг 18](#_Toc92160047)

# 1 ПОСТАНОВКА ЗАДАЧІ

Задача на деформування прямокутної пластини, одна сторона , друга сторона Шарнірно оперта пластина, спочатку ненавантажена пластина

Миттєво навантажується силою, що прикладена в точці, що лежить на діагоналі на відстані 0.25 та 0.5 від сторін. Сила діє протягом 0.05 секунд. Рішити задачу

1) методом К.-Н. Обгрунтувати розбивку

2) Методом Фур’є (Нав’є)

# 2 АНАЛІТИЧНІ РІШЕННЯ

2.1 Виведення основного рівняння

Очевидно, що будь які диференційні рівняння типу рівнянь теорії оболонок неможливо рішити не розуміючи загальної постановки задачі, і як власне отримано загальне рівняння.

Основний параметр – це прогин пластинки 𝑤. Похідна від прогибу по координаті 𝑥 дає величину кута в напрямку абсциси, 𝛾𝑥, а похідна по ординаті – дає величину кута деформування в напрямку осі 𝑦, 𝛾𝑦. Тобто:

Кути деформації пов’язані з кривизнами, які даються формулами:

Де 𝜅𝑥 та 𝜅𝑦 - це кривизни згину, по відповідних напрямках, а 𝜒 - це так звана тут кривизна кручення довільної лінії. Існують відповідності між локальними (локальні – по-перше, вони прикладені до одиниці довжини, а не до всього січення як у балці; по-друге, в кожній точці пластини існують три локальні моменти) згинальними моментами та кривинами. Дальше записуються фізичні рівняння, не зупиняючись на деталях відмітимо, що вони дають зв’язок локальних моментів згину 𝑀𝑥, 𝑀𝑦 та кручення 𝐻 з введеними вище кривинами:

Де 𝐷 -так звана циліндрична жорсткість пластини. З (2.1b) та (2.1c) маємо залежність моментів і переміщень:

Далі записуються три умови рівноваги – дві для локальних моментів, і одна для поперечних сил:

Підставляючи перші два рівняння в третє, отримаємо

Підставивши (2.1d) в (2.1f), отримаємо основне диференційне рівняння теорії пластин:

2.2 Шарнірно закріплена прямокутна пластина

Розглянемо спочатку пластину, що має такі розміри:

Для такої пластини сформулюємо наступні г.у. вздовж сторін перпендикулярних осі 𝑥. Нехай на сторонах 𝑥 = 0 та 𝑥 = 𝑎 пластина є шарнірно опертою, тобто:

Абсолютно аналогічно нехай на краях паралельних осі 𝑥, тобто при 𝑦 = 0 та 𝑦 = 𝑏 пластина також є шарнірно опертою, тобто маємо такі граничні умови:

Граничні умови (2.2b – 2.2e) дозволяють легко обґрунтувати вибір функції переміщень. Очевидно, що якщо взяти:

де 𝑍𝑘,𝑛 -невідомі шукані коефіцієнти, то кожна із таких функцій автоматично задовольняє г.у. Щоб легко отримати рішення (2.1g), треба, щоб функція навантажень також представлялася в подібному вигляді, а саме:

Тоді підставляючи (2.3a) і (2.3b) в основне рівняння (2.1g), отримаємо вираз (2.3с):

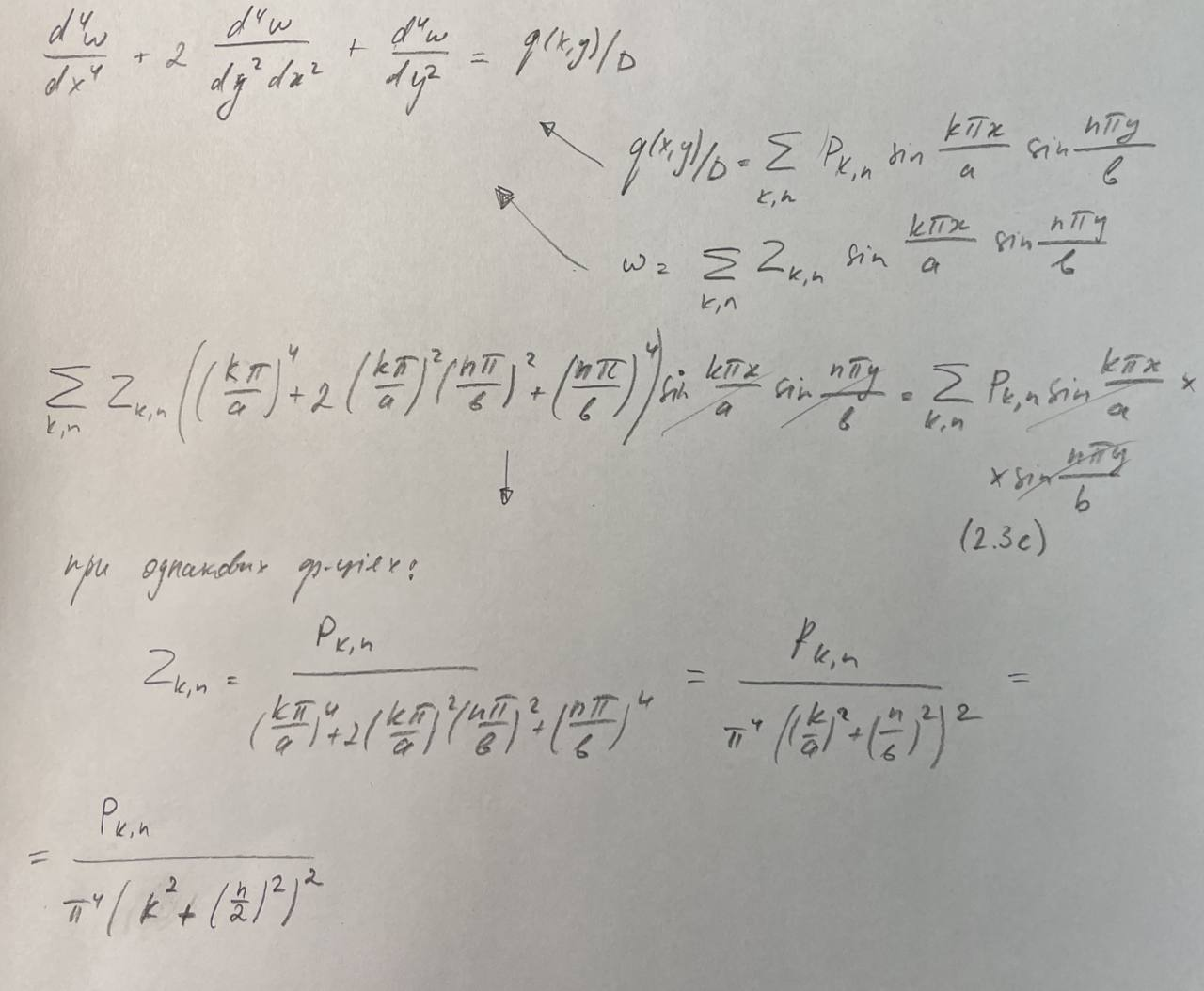


Рис. 2.1

Отримаємо рішенням задачі, адже автоматично задовольняються граничні умови і основне рівняння. Можна сказати, що тут ми йдемо від оберненої задачі. Тобто, якщо навантаження задається в вигляді ряду (2.3b). то рішення буде в формі (2.3а), де коефіцієнти даються (2.3с).

## 2.3 Метод Нав’є

Отримані вище результати дозволяють узагальнити отримані результати для довільного виду навантаження -𝑞(𝑥, 𝑦)/𝐷 = 𝑃(𝑥, 𝑦), Для цього необхідно лише представити це навантаження в вигляді подвійного ряду (2.3b). Це є можливим ще і тому, що система базових функцій, що використовуються, а саме:

є взаємоперпендикулярною, тобто:

Покажемо як довільне навантаження 𝑃(𝑥, 𝑦) розкладається в ряд (2.3b). Для цього помножимо 𝑃(𝑥, 𝑦) на систему функцій (2.4а) і про інтегруємо по всій площі (2.2а). Отримаємо, що коефіцієнти навантаження 𝑃𝑘,𝑛 даються формулою:

Виконаємо заміну:

Тоді (2.4с) буде мати вигляд (2.4е):

З постановки задачі відомо, що точкова зовнішня сила прикладена в точці α, β. Де 𝛿(𝑥) -функція Дірака.

Таким чином враховуючи властивості функції Дірака (рис. 2.5), подвійний інтеграл правої частини (2.4e) можна переписати наступним чином:

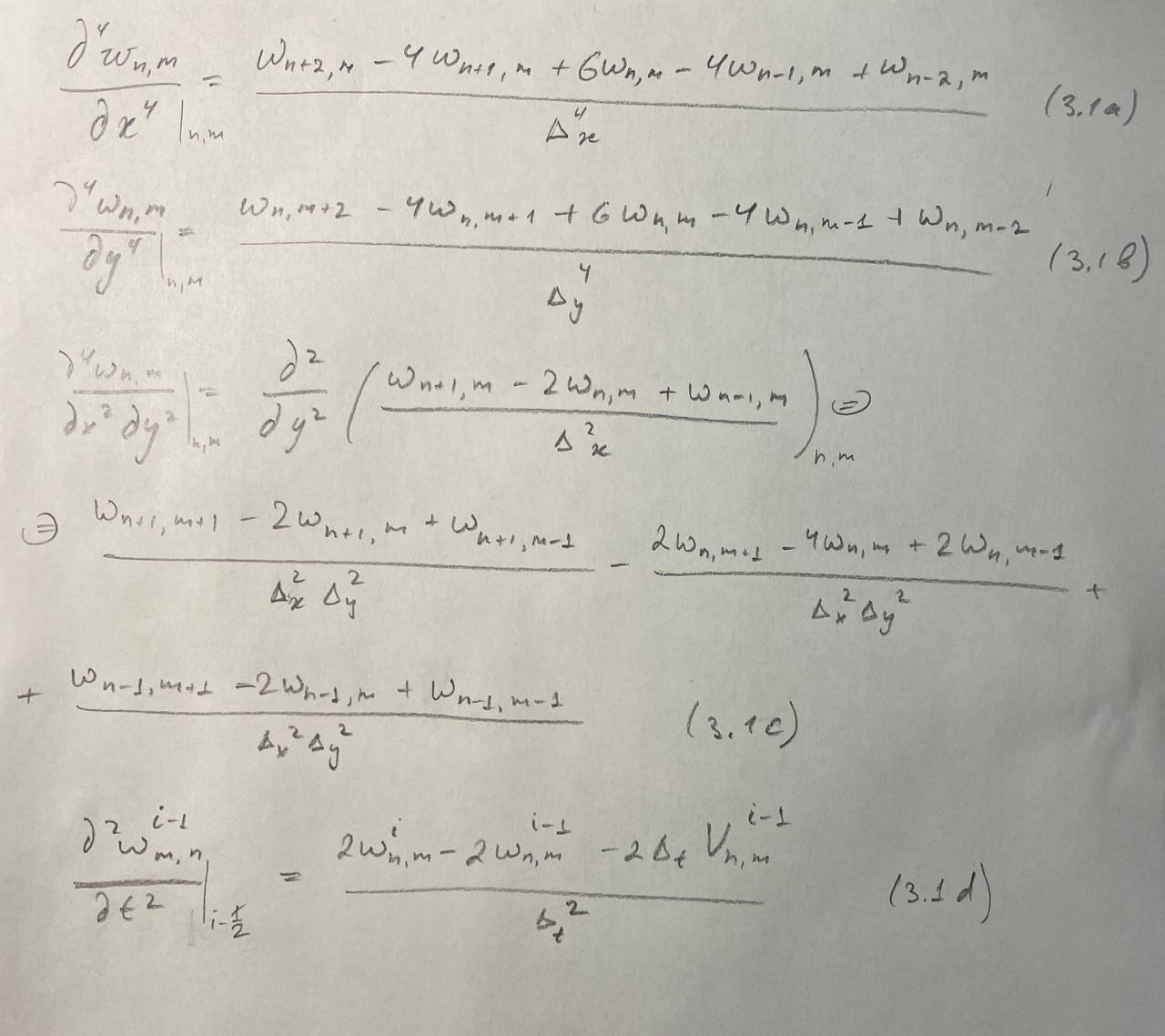
# 3 МЕТОД КРАНКА-НІКОЛСОНА

## 3.1 Різницеві схеми

Для шарнірно опертої пластини граничні умови задаються (2.2b – 2.2e).

Загальний вигляд рівння має наступний вигляд:

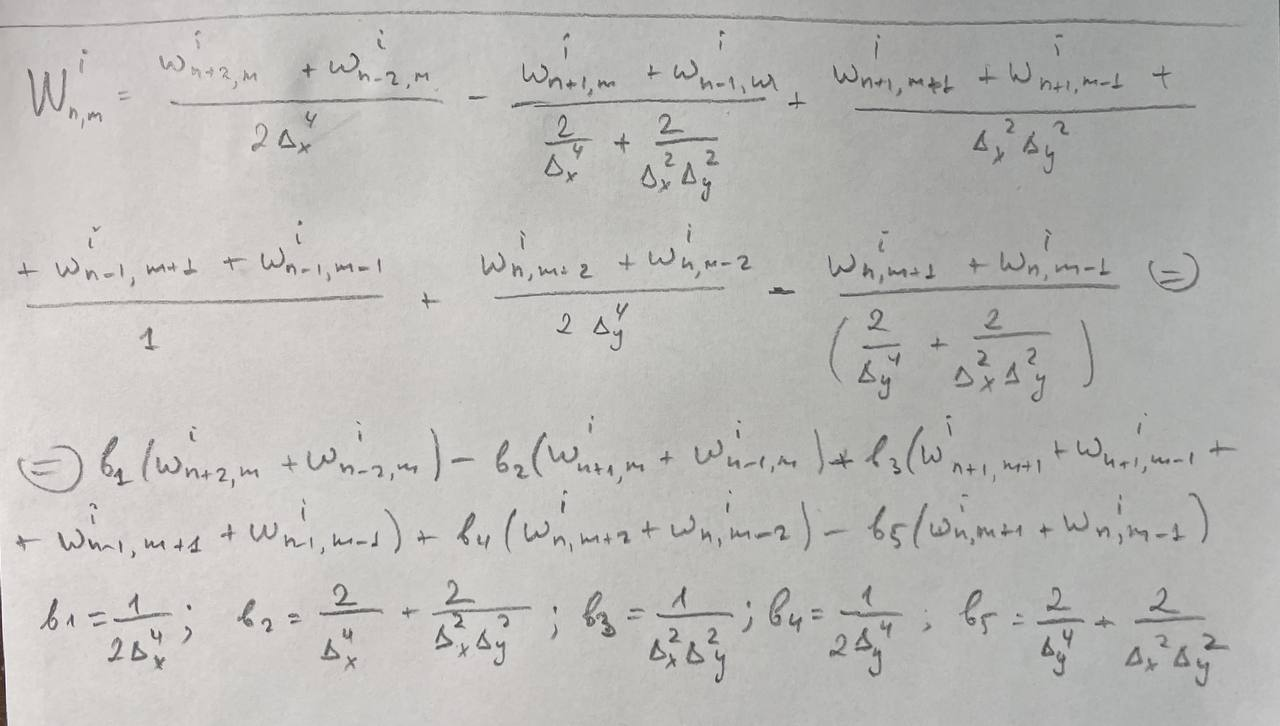
Відповідні різницеві схеми для похідних (3.1a – 3.1d), де індекс n – координата по х, а m – координата по y (рис. 3.1):



Де, так звана швидкість на попередньому моменті часу в точці n, m. Визначається наступним чином:

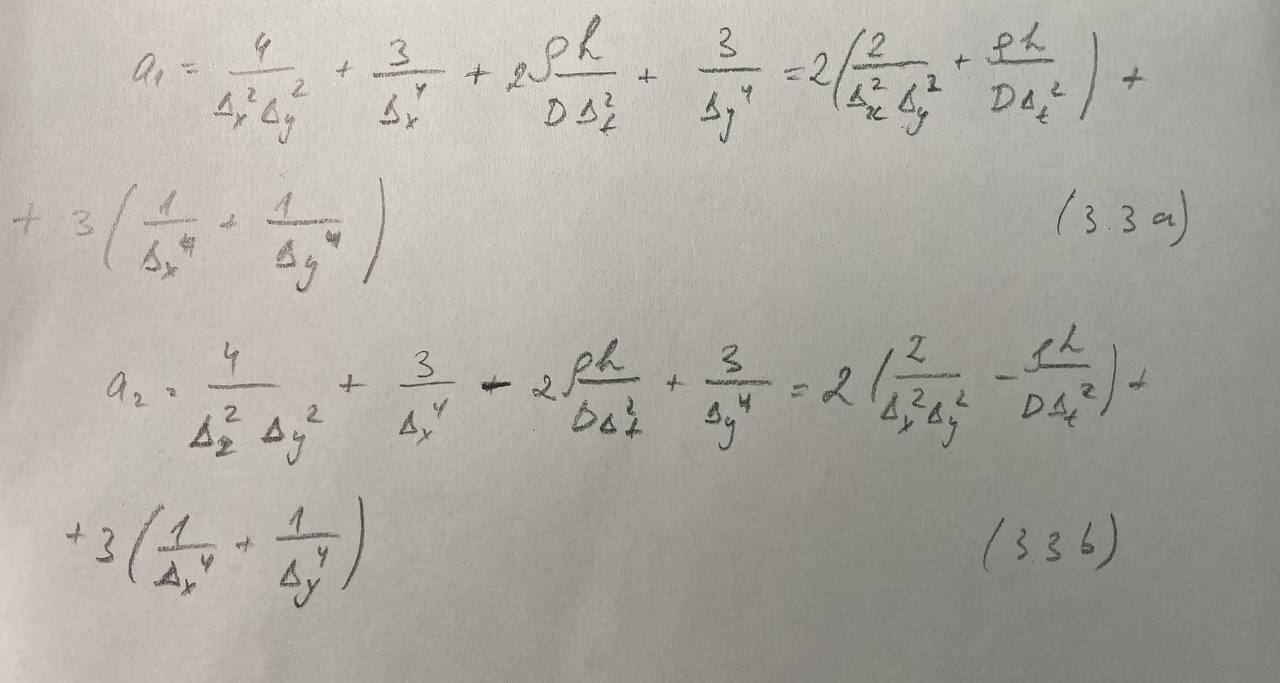
Схема Кранка-Ніколсона передбачає центрування в момент часу , тому перших три доданки (3.1) треба відцентрувати в відповідний момент часу. Таким чаном маємо наступний вираз:

Базуючись на (3.1, 3.1a – 3.1d та 3.2), представимо :



Підставивши всі різницеві схеми 3.1a – 3.1d в 3.2 матимемо наступну схему:

де позначені:



## 3.2 Алгоритм реалізації МКР

Розбиваємо вісь абсцис на N проміжків точками x0, x1,…xN+1 та проводимо через них вертикальні лінії, які розміщені на відстані ∆x = a / N. Аналогічним чином розбиваємо вісь ординат на M проміжків точками y0, y1,…yM+1 та проводимо через них вертикальні лінії, які розміщені на відстані ∆y = b / M.

Додатково вводимо по дві фіктивні змінні (x-1, xN+2) та (у-1, уM+2) через специфіку формулювання граничних умов. Очевидно, що вони знаходяться на прямих:

Таким чином, маємо прямокутник довжиною з N+3 точками, та висотою з M+3. Розглядаючи визначений прямокутник, маємо точок *Zi*, які нумеруються наступним чином:

Всі точки пластини класифікуються за наступними категоріями:

### Група 1.

Кутові точки – це точки з відповідними координатними номерами (−1, −1), (N+1, −1), (−1, M +1) та (N+1, M +1). Для кожної з них вводимо фіктивне рівняння де . Всього для групи А можна скласти 4 рівняння.

### Група 2.

Граничні кутові точки. Вони одночасно належать до двох сторін пластини. В цих точках, формулюються дві г.у.: рівняння *w = 0* та по одному рівнянню виду 3.5 на кожну з двох сторін. Всього 4х3=12 рівнянь. Оскільки пластина закріплена шарнірно, рівняння г.у. має вигляд:

### Група 3.

Точки, що лежать на границях реальної пластини за виключенням кутових. В цих точках ми записуємо по два рівняння виду 3.6b (пластина закріплена шарнірно), а також додаємо умову (тобто по 2 рівняння для кожної точки границі) для сторін паралельних відносно осі x та y відповідно. Таким чином, маємо 4(N−1) + 4(M−1) рівнянь

### Група 4.

Всі внутрішні точки. Загальна кількість яких дорівнює (N−1)(M−1) одиниць. Для кожної такої точки складаємо рівняння вигляду (3.3).

Ці викладки були проведені для фіксованого значення часу , але ми можемо розбити увесь час поки пластина навантажена (у нашому випадку – 0.05 секунди) на T інтервалів часу довжини і вирішувати систему для кожного випадку.

## 3.3 Візуалізація результату

Сила діє на пластину 0.05 с, інтервал зміни часу 0.005, розбиття по N=50, M=100.



|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

При N=50, M=100:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Додаток А. Лістинг

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

import sys

import plotly.graph\_objects as go

import pandas as pd

a, b = 1, 2

r, h, D, P = 1, 1, 1, 0.1

N, M, T = 50, 100, 0.05

dx, dy, dt = a / N, b / M, 0.005

xx = np.linspace(0, a, N + 1)

yy = np.linspace(0, b, M + 1)

print('x interval: {} \ny interval: {} \nt interval: {}'.\

format(dx, dy, dt))

0.05/0.005

w = np.zeros((N+3, M+3))

v = np.zeros((N+3, M+3))

w.shape, v.shape

alpha, beta = 0.25, 0.5

print('x distance: {} \ny distance: {}'.\

format(alpha, beta))

aa = np.logical\_or(xx == xx[xx <= alpha][-1], xx == xx[xx >= alpha][0])

bb = np.logical\_or(yy == yy[yy <= beta][-1], yy == yy[yy >= beta][0])

aa

bb

amask, bmask = np.meshgrid(aa, bb)

powermask = np.zeros((N+3, M+3)).astype(bool)

powermask[1:-1, 1:-1] = np.logical\_and(amask, bmask).T

power = np.zeros((N+3, M+3))

power[powermask] = P / (dx \* dy)

coord = np.zeros((N+3, M+3))

index = lambda n, m: (n + 1) + (m + 1)\*(N + 3)

for i in range(N + 3):

for j in range(M + 3):

coord[i, j] = index(i-1, j-1)

reindex = lambda i: (i % (N + 3) - 1, i // (N + 3) - 1)

a1 = 2 \* (2 / (dx\*\*2 \* dy\*\*2) + r \* h / (D \* dt \*\* 2)) +3 \* (1 / dx \*\* 4 + 1 / dy \*\* 4)

a2 = 2 \* (2 / (dx\*\*2 \* dy\*\*2) - r \* h / (D \* dt \*\* 2)) +3 \* (1 / dx \*\* 4 + 1 / dy \*\* 4)

b1 = 1 / (2 \* dx \*\* 4)

b4 = 1 / (2 \* dy \*\* 4)

b3 = 1 / (dx\*\*2 \* dy \*\* 2)

b2 = 2 / dx \*\* 4 + 2 / (dx \*\* 2 \* dy \*\* 2)

b5 = 2 / dy \*\* 4 + 2 / (dx \*\* 2 \* dy \*\* 2)

def wmatrix(i, j):

xexpr = np.zeros(((N + 3) \* (M + 3)))

xexpr[index(i+2, j)] = b1

xexpr[index(i-2, j)] = b1

xexpr[index(i+1, j)] = -b2

xexpr[index(i-1, j)] = -b2

for n, m in [(i+1, j+1), (i+1, j-1), (i-1, j+1), (i-1, j-1)]:

xexpr[index(n, m)] = b3

xexpr[index(i, j+2)] = b4

xexpr[index(i, j-2)] = b4

xexpr[index(i, j+1)] = -b5

xexpr[index(i, j-1)] = -b5

return xexpr

def dpoints(pw, pv, power):

bexpr = np.zeros(((N-1)\*(M-1)))

xexpr = np.zeros(((N-1)\*(M-1), (N + 3) \* (M + 3)))

wconst = lambda n, m: b1 \* (pw[n+2, m] + pw[n-2,m]) -\

b2 \* (pw[n+1, m] + pw[n-1,m]) +\

b3 \* (pw[n+1, m+1] + pw[n+1, m-1] + pw[n-1, m+1] + pw[n-1, m-1]) +\

b4 \* (pw[n, m+2] + pw[n, m-2]) -\

b5 \* (pw[n, m+1] + w[n,m-1])

for ind1, i in enumerate(range(1, N)):

for ind2, j in enumerate(range(1, M)):

xexpr[(M-1)\*ind1 + ind2, :] = wmatrix(i, j)

xexpr[(M-1)\*ind1 + ind2, index(i, j)] = a1

bexpr[(M-1)\*ind1 + ind2] -= (wconst(i, j) + a2 \* pw[i, j])

bexpr[(M-1)\*ind1 + ind2] += 2 \* r \* h / (D \* dt) \* pv[i, j] + power[i, j] / D

return xexpr, bexpr.reshape((N-1)\*(M-1), -1)

axd, bd = dpoints(w, v, power)

axd.shape, bd.shape

def cpoints(N, M):

xexpr = np.zeros((4 \* (N - 1), (N + 3) \* (M + 3)))

for ind1, j in enumerate([0, M]):

for ind2, i in enumerate(range(1, N)):

qindex = 2\*(N-1) \* ind1 + 2 \* ind2

#print((i, j, index(i, j)), qindex, qindex + 1)

# г.у 3.21 (1)

#print(qindex, (i, j, index(i, j)))

xexpr[qindex, index(i, j)] = 1

# г.у 3.21 (2)

#print(qindex + 1, (i, j, index(i, j-1)),

# (i, j, index(i, j+1)))

# xexpr[qindex + 1, index(i, j-1)] = 1 / (2 \* dy)

# xexpr[qindex + 1, index(i, j+1)] = 1 / (2 \* dy)

xexpr[qindex+1, index(i,j-1)] = 1 / dy \*\* 2

xexpr[qindex+1, index(i, j)] = -2 / dy \*\* 2

xexpr[qindex+1, index(i,j+1)] = 1 / dy \*\* 2

#print()

yexpr = np.zeros((4 \* (M - 1), (N + 3) \* (M + 3)))

for ind1, i in enumerate([0, N]):

for ind2, j in enumerate(range(1, M)):

qindex = 2\*(M-1) \* ind1 + 2\*ind2

#print((i, j, index(i, j)), qindex, qindex + 1)

# г.у 3.20 (1)

#print(qindex, (i, j, index(i, j)))

yexpr[qindex, index(i, j)] = 1

# г.у 3.20 (2)

#print(qindex+1, (i, j, index(i-1, j)), (i, j, index(i+1, j)))

yexpr[qindex+1, index(i+1,j)] = 1 / dx \*\* 2

yexpr[qindex+1, index(i, j)] = -2 / dx \*\* 2

yexpr[qindex+1, index(i-1,j)] = 1 / dx \*\* 2

return np.vstack((xexpr, yexpr)), np.zeros(4\*(N - 1) + 4\*(M - 1)).reshape(4\*(N - 1) + 4\*(M - 1), -1)

axc, bc = cpoints(N, M)

axc.shape, bc.shape

axc, bc = cpoints(N, M)

axc.shape, bc.shape

#axc[15].reshape(M+3, -1).T

def bpoints(N, M):

expr = np.zeros((12, (N + 3) \* (M + 3)))

for ind, (i, j) in enumerate([(0, 0), (0, M),

(N, 0), (N,M)]):

#print(3 \* ind, i, j, index(i, j))

expr[3 \* ind, index(i, j)] = 1

# г.у. (3.18)

#print(3 \* ind + 1, index(i, j-1), index(i, j+1))

# expr[3 \* ind + 1, index(i, j-1)] = 1 / (2 \* dy)

# expr[3 \* ind + 1, index(i, j+1)] = 1 / (2 \* dy)

expr[3 \* ind + 1, index(i, j-1)] = 1 / dy \*\* 2

expr[3 \* ind + 1, index(i, j) ] = -2 / dy \*\* 2

expr[3 \* ind + 1, index(i, j+1)] = 1 / dy \*\* 2

# г.у. (3.19)

#print(3 \* ind + 2, index(i+1, j), index(i-1, j))

expr[3 \* ind + 2, index(i+1, j)] = 1 / dx \*\* 2

expr[3 \* ind + 2, index(i, j) ] = -2 / dx \*\* 2

expr[3 \* ind + 2, index(i-1, j)] = 1 / dx \*\* 2

return expr, np.zeros(12).reshape(12, -1)

axb, bb = bpoints(N, M)

axb.shape, bb.shape

axb, bb = bpoints(N, M)

axb.shape, bb.shape

#axb[2].reshape(M + 3, -1).T

def apoints(N, M):

expr = np.zeros((4, (N + 3) \* (M + 3)))

for ind, (i, j) in enumerate([(-1, -1 ), (-1, M+1),

(N+1, -1), (N+1,M+1)]):

#print(ind, i, j, index(i, j))

expr[ind, index(i, j)] = 1

return expr, np.zeros(4).reshape(4, -1)

axa, ba = apoints(N, M)

axa.shape, ba.shape

# Перевірка 1

#axa[2].reshape(M+3, -1).T

# Перевірка 2

for i in range(axa[0].shape[0]):

if axa[3, i] == 1:

print(i, reindex(i))

A = np.vstack((axa, axb, axc, axd))

B = np.vstack((ba, bb, bc, bd))

A.shape, B.shape

res = np.linalg.solve(A, B)

fig, ax = plt.subplots(1, 1, figsize=(10,12))

X, Y = np.meshgrid(xx, yy)

Z = res.reshape(M + 3, -1)[1:-1, 1:-1]

norm = matplotlib.colors.Normalize(-abs(Z).max(), abs(Z).max())

p = ax.pcolor(X, Y, Z, norm=norm, cmap=matplotlib.cm.Spectral\_r, shading='auto')

ax.axis('tight')

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

# ax.grid()

# ax.xaxis.set\_major\_locator(matplotlib.ticker.MaxNLocator(4))

# ax.yaxis.set\_major\_locator(matplotlib.ticker.MaxNLocator(4))

cb = fig.colorbar(p, ax=ax)

cb.set\_label('Z')

# cb.set\_ticks([-1, -.5, 0, .5, 1])

X, Y = np.meshgrid(xx, yy)

Z = res.reshape(M + 3, -1)[1:-1, 1:-1]

fig = go.Figure(data=[go.Surface(z=Z, x=X, y=Y)])

fig.update\_layout(title='Surface', autosize=False,

width=950, height=950,

margin=dict(l=65, r=50, b=65, t=90))

fig.show()

def update\_v(ow, nw, v):

return 2 \* (nw - ow) - v

current\_power = power

current\_w, current\_v = w, v

n\_iter, data = 60, []

for i in range(n\_iter):

aequations, aconst = apoints(N, M)

bequations, bconst = bpoints(N, M)

cequations, cconst = cpoints(N, M)

if i \* dt > T:

current\_power = np.zeros\_like(power)

dequations, dconst = dpoints(current\_w, current\_v,

current\_power)

A = np.vstack((aequations, bequations, cequations, dequations))

B = np.vstack((aconst, bconst, cconst, dconst))

z = np.linalg.solve(A, B)

fig, ax = plt.subplots(1, 1, figsize=(7, 9))

X, Y = np.meshgrid(xx, yy)

Z = z.reshape(M + 3, -1)[1:-1, 1:-1]

if i%2 != 0:

Z= -Z

norm = matplotlib.colors.Normalize(-abs(Z).max(), abs(Z).max())

p = ax.pcolor(X, Y, Z, norm=norm, cmap=matplotlib.cm.Spectral\_r, shading='auto')

ax.axis('tight')

ax.set\_title('Time: {}'.format(i \* dt))

ax.set\_xlabel(r"x", fontsize=18)

ax.set\_ylabel(r"y", fontsize=18)

ax.xaxis.set\_major\_locator(matplotlib.ticker.MaxNLocator(4))

ax.yaxis.set\_major\_locator(matplotlib.ticker.MaxNLocator(4))

cb = fig.colorbar(p, ax=ax)

cb.set\_ticks([-1, -.5, 0, .5, 1])

plt.show()

# fig, ax = plt.subplots(1, 1, figsize=(16, 12),

# subplot\_kw={'projection': '3d'})

# def title\_and\_labels(ax, title):

# ax.set\_title(title)

# ax.set\_xlabel("$x$", fontsize=16)

# ax.set\_ylabel("$y$", fontsize=16)

# p = ax.plot\_surface(X, Y, Z, rstride=1, cstride=1,

# linewidth=0, antialiased=False, norm=norm, cmap=mpl.cm.Blues)

# cb = fig.colorbar(p, ax=ax, shrink=0.6)

# title\_and\_labels(ax, "plot\_surface")

current\_v = update\_v(current\_w, z.reshape(M + 3, -1).T, current\_v)

current\_w = z.reshape(M + 3, -1).T

data.append(current\_w)